UNLIMITED
UNCLASSIFIED

ILLIMITÉE
NON CLASSIFIÉE

Technical Report

Rapport technique

1988/11

TR-SYS-019
NRC NO. 29977

GENERAL INFERENCE ENGINE FOR EXPERT SYSTEMS: USER'S GUIDE

S. Moran

Division of
Mechanical Engineering

Division de
génie mécanique

National Research Council Canada
Conseil national de recherches Canada

Canada

89-03318/

ISION OF MECHANICAL ENGINEERING PUBLICATIONS

DM **(Division of Mechanical Engineering Report)**
Scientific and technical information considered important, complete and a lasting contribution to existing knowledge.

TR **(Technical Report)**
Information less broad in scope but a substantial contribution to existing knowledge.

CTR **(Controlled/Classified Technical Report)**
A Technical Report with controlled distribution for national security, proprietary or other reasons.

LM **(Laboratory Memorandum)**
Preliminary or exploratory information with controlled distribution.

CAT **(Calibration Analysis and Test Report)**
Information on minor laboratory projects or services.

PUBLICATIONS DE LA DIVISION DE GÉNIE MÉCANIQUE

DM **(Rapport de la Division de génie mécanique)**
Informations scientifiques et techniques jugées importantes, complètes et susceptibles de contribuer de façon durable à l'avancement des connaissances courantes.

TR **(Rapport technique)**
Informations de moindre importance, mais pouvant contribuer substantiellement à l'avancement des connaissances actuelles.

CTR **(Rapport technique à diffusion contrôlé/classifié)**
Rapport technique à diffusion contrôlée pour des raisons de sécurité nationale, de propriété intellectuelle et autres.

LM **(Mémoire de laboratoire)**
Informations préliminaires ou de nature exploratoire à diffusion contrôlée.

CAT **(Rapport d'étalonnage d'analyse et d'essai)**
Informations sur de petits projets ou des services de laboratoire.

UNLIMITED
UNCLASSIFIED

ILLIMITÉE
NON CLASSIFIÉE

GENERAL INFERENCE ENGINE FOR EXPERT SYSTEMS: USER'S GUIDE

MOTEUR D'INFÉRENCE GENERAL POUR SYSTÈMES EXPERTS: GUIDE D'APPLICATION

S. Moran

Technical Report

Rapport technique

1988/11

TR-SYS-019
NRC No. 29977

V. Thomson, Head/Chef
Systems Laboratory
Laboratoire des systèmes

J. Ploeg
Director/
Directeur

ABSTRACT

Of the technologies designated by artificial intelligence, that of expert or knowledge-based systems is the most advanced in terms of practical applications. An expert system is composed of a knowledge or rule base and a software module called an inference engine which applies a knowledge base to make judgements. This report describes an inference engine implemented in PROLOG with capabilities to explain its reasoning, to accommodate uncertainties and to manage the knowledge base it uses for reasoning.

RÉSUMÉ

De toutes les technologies designées par intelligence artificielle, celle de systèmes experts ou systèmes à base de connaissances est la plus avancée en termes d'applications pratiques. Un système expert comprenne une base des connaissances ou des réglements et un logiciel appelé moteur d'inférence avec le but d'utiliser ces connaissances pour faire des jugements ce rapport décrit un moteur d'inférence exécuté utilisant le langage de programmation PROLOG avec les capacités d'expliquer son raisonnement, d'accommoder les incertitudes et d'entretenir la base de connaissances utilisée pour raisonner.

CONTENTS

		Page
	ABSTRACT/RÉSUMÉ	(i)
1.0	INTRODUCTION	1
2.0	THE GIEES INFERENCE ENGINE	1
	2.1 Introduction	1
	2.2 Commands Description	2
	2.2.1 The add command	3
	2.2.2 The cf command	4
	2.2.3 The delete command	4
	2.2.4 The edit command	5
	2.2.5 The exit command	5
	2.2.6 The go command	5
	2.2.7 The help command	7
	2.2.8 The how command	7
	2.2.9 The list command	8
	2.2.10 The quiet command	8
	2.2.11 The read command	8
	2.2.12 The status command	9
	2.2.13 The talk command	9
	2.2.14 The why command	9
	2.2.15 The write command	10
	2.3 Start-Up Procedure	10
3.0	THE DEVELOPMENT OF A KNOWLEDGE BASE	10
	3.1 Introduction	10
	3.2 Certainty Factors	10
	3.3 Attribute Description	13
	3.4 Rule Description	15
	3.5 Meta Rules and Facts	16
	3.6 Example of a Knowledge Base	18
4.0	APPLICATIONS	18
5.0	FUTURE PLANS	19
6.0	BIBLIOGRAPHY	19
	DOCUMENTATION PAGE	

GENERAL INFERENCE ENGINE FOR EXPERT SYSTEMS: USER'S GUIDE

1.0 INTRODUCTION

With the development of artificial intelligence (AI) techniques, scientists of all types have attempted to apply them in decision-making situations. These situations often require use of heuristic rules as opposed to algorithms. Expert systems are one of the first products of AI research that partially fulfills these expectations. Though much work is still to be done, some expert systems have already proven useful in widely different fields and applications.

For example, consider the problem of the selection of mining equipment (Hadjigeorgiou 1988). These selection decisions involve a number of criteria and associated parameters. The exact influence of these parameters on the criteria are often not well understood. Still, experts are able to make good choices of equipment. How do they do it? Their knowledge is often acquired through experience and is generally expressable in the form of rules. For example, an engineer might say that "If such a condition or conditions exists, then this is probably the best type of equipment to use." This type of expertise is not necessarily rigorous nor all-encompassing and often leads to uncertain conclusions.

Since the number of experts with this advanced experience and knowledge is small, the goal of workers in the field of expert systems is to replicate this knowledge in rule bases and databases and to use it when these experts are not available. Use of such expert systems have two important side effects: (1) they help other persons acquire the expertise because these systems are able to explain their chain of reasoning while inferring new facts, and (2) they can replace the expert when one is not available, possibly to confirm a result obtained by some other method or to provide a starting point and reduce the time spent on a project.

2.0 THE GIBBS INFERENCE ENGINE

2.1 Introduction

An expert system has two main components. The first component is the inference engine (IE), which provides the reasoning power used to manipulate the knowledge base and can, with some input from the user, provide or deduce an answer for the goal of the consultation. The second part is the knowledge base (KB) where the expertise, in a particular domain, is represented in the form of base facts and "IF ... THEN ..." rules.

The two main data structures used by the IE are the goal stack (GS) and the working storage (WS). The GS is a stack structure that contains the goals and subgoals which the IE must infer by using the rule base or querying the user. WS contains facts that have been inferred or given by the user. A rule is associated with each fact that was used to infer the fact (code 0 if given by user and -1 if a base fact) and a certainty factor (CF) is used to represent the certainty with which the fact can be asserted.

The General Inference Engine for Expert Systems (GIEES) is a rule-based inference engine written in PROLOG and, since it provides many of the components for the development and use of an expert system, it could be called an expert system shell. The next section describes the command language of the system, while the third section shows how to develop a KB usable by the inference engine.

GIEES comprises six modules for

(1) command interpretation and dispatching
(2) reasoning or inferencing
(3) manipulation of knowledge bases
(4) explanation of reasoning to the user
(5) general support functions (library), and
(6) user help

2.2 Commands Description

When the program is run, the screen is cleared, a welcome message is displayed, and the system prompt (>) appears. GIEES can nov accept commands. The system commands are divided in four main categories and are described in this section. The four categories are inference engine commands, knowledge base manipulation commands, explanation commands, and miscellaneous commands. A list of all commands within each category is as follows.

Inference engine commands:

- go	- begins a consultation session
- cf premise	- changes the current premise algebra
- cf conclusion	- changes the current conclusion algebra
- cf fact	- changes the current fact algebra

Knowledge base commands:

- add fact	- adds a value to an attribute in the WS
- delete fact	- deletes a fact from the WS

- list fact	- lists facts from WS to the screen
- list rule	- lists the rules to the screen
- list attribute	- lists the attributes to the screen
- list conclusion	- lists the last session's result
- write fact	- same as list except output to a file
- write rule	- same as list except output to a file
- write attribute	- same as list except output to a file
- write how	- writes all facts in the WS with the rules used to infer their values

Explanation commands:

- how	- explains how an attribute value is inferred
- why	- explains why a question is being asked

Miscellaneous:

- help	- lists all valid commands
- read	- reads in a KB
- edit	- opens a window to modify a KB
- talk	- starts a trace of the rule evaluation and inferences
- quiet	- stops the trace
- status	- prints the available memory and current CF algebras
- exit	- returns to the operating system.

2.2.1 The add command

This command is used to add base facts to WS. It is used primarily to observe the reactions of the expert system when different values are assigned to an attribute. The attribute must first be declared in the KB before it can be given a value with the add command.

Example:

Given the following KB declaration,

```
es_attribute (10,"interest_rate","real","interest rate","%").
```

we could execute the following command:

```
> add interest_rate is 9.25
```

2.2.2 The cf command

This command changes the algebra used to calculate the certainty factor (CF) associated with the value of an attribute. At different stages of a consultation, different algebras can be used to combine these certainty factors. The reader is referred to Chapter 10 of Holsapple (1986) for a description and justification of each of the algebras and their effect on the IE. There are three categories of certainty factors: the certainty of a premise, the certainty of a rule conclusion, and the certainty of a fact. These different factors can also be combined using various calculation methods.

Syntax:

cf <cf_category> <cf_algebra>

where

<cf_category> = premise, conclusion or fact

If <cf_category> is a premise or conclusion then

<cf_algebra> = Minimum, Product, Average or Bonczek-Eagin

else if <cf_category> = fact, then

<cf_algebra> = Maximum, Probability Sum, Average or Bonczek-Eagin.

Examples:

```
cf premise product
cf premise average
cf fact
cf conclusion Bonczek-Eagin
```

Note: If the user does not specify the algebra on the command line, the default value for the specified category is used (see Section 3.2).

2.2.3 The delete command

The delete command is used to delete base facts from the KB during the development phase of the expert system. As in the case of the add command it is used to study the reactions of the system when certain facts are deleted from the KB.

Syntax:

delete <attribute name>

Example:

Given the following KB declaration

es_attribute (10,"interest_rate","real","interest rate","%").

we could execute the following command:

> delete interest_rate

2.2.4 The edit command

During development of an ES, the text describing the KB is often modified. To facilitate the updating and testing of a KB, an edit command was incorporated in the system. When the editor is called, a window is opened and the text of the KB is ready for editing. The Turbo PROLOG editor is used and the reader is referred to the user's manual (Borland). When the editing is complete, the user returns to GIEES by pressing the <ESC> key. If changes were made to the KB, the file is updated and the KB in memory is replaced by the newly modified KB. If no changes were made to the KB then no action is taken. An extension (.pro) is added automatically to the KB name to create the file name.

Syntax:

edit <KB name>

2.2.5 The exit command

The exit command terminates execution of the GIEES system and returns control to the operating system.

2.2.6 The go command

The go command starts a consultation session using the IE. The main goal of the consultation may be defined in two ways. One method is to specify the name of the attribute for which the user seeks a value on the command line invoking the IE.

Example:

> go loading_equipment

This command starts the IE and the main goal of the consultation will be to find a value for the attribute loading_equipment. The second method is to add the main goal in the KB by adding the declaration giees_goal along with the name of the attribute.

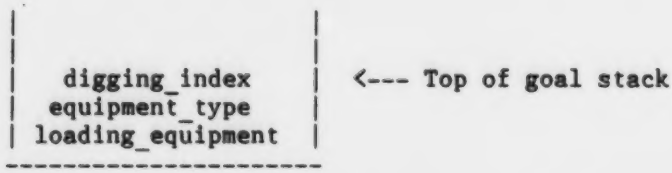
Example:

```
giees_goal (loading_equipment)
```

This declaration indicates that the main goal for this KB is to find a value for the attribute loading_equipment. We will see later how to create a KB (Section 3.5). However, note that any attribute mentioned on the command line has priority on any goal defined in the KB. GIEES first checks the command line for the main goal. If no goal is present, it then verifies the KB. If no goals are declared in the KB, GIEES generates an error message. The goal in the KB acts as a default main goal. The command line can contain only one main goal while the KB may contain as many as the user wants.

Once the main goal is identified, it is placed on the goal stack (GS) and evaluation begins. The GS contains the list of all attributes or subgoals that must have a value before the IE can execute a particular rule. In effect, the GS represents the chain of reasoning used to infer a value for the main goal. If it is not possible to find a value for the top subgoal of the GS, the IE is blocked and must stop without finding a value for the main goals.

Example:



In this example, GIEES must first find a value for the digging_index before it can infer the type of equipment necessary (loader, excavator, etc.). Once GIEES finds a value for the digging_index attribute, it is removed from the GS and the IE proceeds to the equipment_type subgoal, and so on. The consultation is completed when the goal stack is empty or when no more rules permit evaluation of the current goal (i.e., the top goal of the GS).

Evaluation of goals is a three-step process. The first step is to find a rule to evaluate. If a rule is found, it is evaluated and the result (true, false or unknown) is returned. The last step modifies the environment (GS and WS) according to the result of the evaluation.

To evaluate a goal, the IE uses the backward chaining method of inference. It examines the conclusions of each rule and extracts those rules whose conclusions assign a value to the current subgoal. From that subset of rules, the IE chooses the first one (the order of the rules in the KB influence the order in which the rules are evaluated) and tries to evaluate the rule's conditions. These conditions are statements that are true, false, or unknown. If all conditions are true, then the rule is said to be true. If only one of them is false, it is said to be false.

If the value of an attribute is not known in the present context, its value is said to be unknown and the attribute is pushed on the GS. It becomes the current goal to be evaluated before the IE can continue evaluation of the original goal. When this subgoal is evaluated and a value has been inferred for the attribute, the IE can then return to evaluation of the original rule to verify if the newly found value of the attribute is compatible with the condition of the original rule. Note that, before evaluating a rule, the IE checks that none of the other conditions of that rule are false. This check prevents the IE from evaluating the other conditions which have an unknown value and when it knows that the rule will evaluate to false anyway.

The last step is to modify the environment (GS and WS) according to the value of the evaluated rule. If the rule is false, the rule is marked as rejected and the search continues for the next rule (step 1). If the rule is unknown (i.e., the value of one of the attributes in the conditions is not known), then that attribute is pushed on the GS and becomes the current goal. If the rule is true then the conclusions of that rule are evaluated and the attribute's values are added to the list of inferred facts, the top goal from the GS is removed, the rule that placed the attribute on the GS is reactivated and the evaluation starts over again until one of the stop conditions occurs.

Evaluation stops when the GS is empty (successful termination) or when there are no more rules with which to infer the current goal (failure termination). In the case of failure termination, the IE has one more option before quitting. The current subgoal (attribute) might have the askable property. This property tells the IE that it can ask the user to enter a value for that attribute in the event that the IE cannot infer a value using the rule set. After the user has answered, the same process of updating the data structures is executed, as if a rule had been evaluated to true. The evaluation then continues as before.

2.2.7 The help command

This command displays the list of all commands and a brief description of each. If the command is followed by the name of a GIBBS command then a more detailed description of that particular command is displayed.

2.2.8 The how command

Whenever a value has been found for an attribute, the IE keeps the number of the rule used to infer its value. This allows the system to explain how the value of an attribute was inferred. By printing the rule that permitted the IE to infer the value, the user can verify the conditions which permitted deduction of the current value for that attribute.

Example:

```
> how loading_equipment
```

This command will display the rules the IE used to obtain the value of the attribute loading_equipment. If the value of the attribute was given by the user, if it has no value, or if the attribute does not exist, then GIEES will display a message to that effect.

2.2.9 The list command

The list command is used to list different parts of the KB on the display screen. Four KB elements can be viewed: facts, rules, attributes and conclusions. The legal list commands and their description follovs.

```
> list fact
```

This command is called usually after a consultation to list all facts that vere inferred during the last consultation. All inferred facts are deleted at the begining of each nev consultation, so that the list only includes those inferred during the last consultation session.

```
> list fact base
```

Certain facts are designated base facts because they are included in the KB and represent given or base knovledge.

```
> list rule [n]
```

This command will list the rule specified by the parameter n. If no rule number is given then all rules from the KB will be displayed.

```
> list attribute
```

This command will list all attributes defined in the KB. The command displays the attribute number, type, name and list of its legal values.

2.2.10 The quiet command

This command switches off the trace and suspends further trace displays.

2.2.11 The read command

The KB used by GIEES must be loaded in main memory before the start of a consultation. While different KBs can be loaded, only one KB can be in main memory at one time. If the user wishes to merge two KBs, it can be done using the edit command to create a single KB from the two KBs. The resulting KB can then be loaded in main memory.

The read command is used to load a KB into main memory. An extension (.pro) is added automatically to the name of the KB to create the file name.

Syntax:

read <KB name>

Example:

read mine_kb

2.2.12 The status command

This commands displays the current status of the GIEES system. It displays the date and time, the amount of memory available to the system, and the current CF algebras for each of the possible CF combination type (premise, conclusion, or fact).

2.2.13 The talk command

The talk command activates the trace switch. Whenever the trace switch is on, the IE will display the value (true, false, or unknown) of all the rules that are evaluated. It will also display the value of all attributes as they are inferred during the consultation. The value of the attribute is followed by two numbers. The first number is the number of the rule that inferred the value (-1 for user input) and the second number is the CF associated with this value.

2.2.14 The why command

The why command can be used during a consultation whenever the IE asks the user a question. In response, the IE will display the reason for asking the question in the following form.

If we can prove that

{conditions of the rule being evaluated}

we will then be able to infer that

{conclusions of the rule}

where one of the conclusions is the value sought for the attribute which, in this case, is the top element in the goal stack. All conditions that have been inferred in some way (user input or IE inference) are followed by the string *** proved ***. This command can only be used during the consultation when the IE asks the user a question. A null answer (i.e, a carriage return) will result in the display of the definition (if one is provided in the KB) of the attribute whose value is being sought along with the legal values that can be assigned to this attribute.

2.2.15 The write command

The write command is similar to the list command except that the output is directed to a file which can then be printed.

2.3 Start-Up Procedure

The file containing the IE shell is named es.exe. To initiate execution of the IE (also called GIEES), the user enters ES (representing expert system) at the DOS prompt followed by a carriage return.

A> ES <CR>

GIEES will then display a welcome message followed by the GIEES system prompt which is >. GIEES is then ready to accept the user's commands.

3.0 DEVELOPMENT OF A KNOWLEDGE BASE

3.1 Introduction

By itself, GIEES is not an ES. An ES is the combination of GIEES with a knowledge base. Design of an expert system is a long and difficult task, requiring the help of many people with different areas of expertise. The two most important persons are the knowledge engineer and the expert in the domain of interest. The expert brings the knowledge required to make decisions in most situations arising in that domain. The knowledge engineer compiles the knowledge of the expert in the form of a coherent set of rules that can be used efficiently and without redundancy.

In our context, a KB is defined in the form of a series of Turbo Prolog statements. The KB is divided in four sections: a declaration section, an attribute section, a rule section and a base fact section. The syntax is the same as that used to define prolog facts and is compatible with the 'readterm' predicate of Turbo PROLOG. When defining a KB, each statement must be on a single line (i.e, a statement cannot span multiple lines). Also, all KB strings must be double-quoted. All identifiers and words must be in lowercase. All identifiers and words that must have uppercase letters or blanks (such as the description of an attribute) must be surrounded by double quotes. Case is significant in identifiers, so be careful when you use this double quote format.

3.2 Certainty Factors

An important capability of an expert system is that of manipulating uncertain data. The degree of uncertainty of a fact included in the KB and WS is expressed by certainty factors. In GIEES, these values are expressed by an integer value ranging from 0 to 100, where 100 is complete certainty and 0 indicates absolute falseness.

In evaluation of a rule, each condition is either true, false, or unknown. In evaluating the conditions of a rule, a fact with a CF of 15 or more is considered true; otherwise, it is false. This is the default threshold value and can be changed by a new definition in the declaration section of a KB. For example, if a rule is found to be true, (i.e., all conditions or premises of that rule are true (CF greater than 15)), we must first combine the relative certainty (CF) of each premise. One of the methods used to combine these CF values is to use the smallest CF. For example, if the CF values of the associated conditions are 35, 55, and 100, GIEES could use the value 35 as the combined CF for the premise.

The relative certainty of the premise also influences the certainty of the inference so we must combine the CF from the premise with the CF of the conclusion. One method used by GIEES is the product formula where the CF of the premise is multiplied by the CF of the conclusion.

```
NewCF = (PremiseCF * ConclusionCF) / 100
```

where

```
NewCF        = new CF for the conclusion being inferred
PremiseCF    = combined CF of all premises
ConclusionCF = CF of the conclusion when all premises have a CF of
               100.
```

Example:

```
PremiseCF    = 35
ConclusionCF = 80

NewCF = (35 * 80) / 100
      = 28
```

In situations where two or more rules conclude with the same value for a fact but with a different CF, these CF values must then be combined to determine the CF for that fact. A formula used to calculate this new CF is the probability sum formula.

```
NewCF = (CF1 + CF2) - ((CF1 * CF2)) / 100)
```

where

```
NewCF = The new combined CF of the fact
CF1   = The CF from the first rule which inferred this fact
CF2   = The CF from the second rule
```

Example:

```
CF1 = 80
CF2 = 65

NewCF = (80 + 65) - ((80 * 65) / 100)
      = 145 - 52
      = 93
```

For the premise and conclusion, the user can choose among four formulas or algebras:

(1) minimum (default algebra for premises)
(2) product (default algebra for conclusions)
(3) average
(4) Bonczek-Eagin

The minimum algebra tends to be the most pessimistic method while the product algebra is the most optimistic.

Minimum:

```
NewCF = MIN (CF1, CF2)
```

Product:

```
NewCF = CF1 * CF2 / 100
```

Average:

```
NewCf = (MIN (CF1, CF2) + ((CF1 * CF2) / 100)) / 2
```

Bonczek-Eagin:

```
NewCF = ((CF1 * CF2) / 100) * (2  (MAX (CF1, CF2) / 100))
```

For the facts, the user can choose among four formulas or algebras:

(1) maximum
(2) probability sum (default algebra for facts)
(3) fact average
(4) fact Bonczek-Eagin

Maximum:

```
NewCF = MAX (CF1, CF2)
```

Probability sum:

NewCF = (CF1 + CF2) - ((CF1 * CF2) / 100)

Fact average:

NewCf = (MAX (CF1, CF2) + (CF1 + CF2) ((CF1 * CF2) / 100)) /2

Bonczek-Eagin:

NewCF = MAX(CF1, CF2)+((CF1*CF2)/100*(1(MAX(CF1,CF2)/100)))

3.3 Attribute Description

An attribute is an element of knowledge that has several characteristics. These characteristics include an attribute number (unique to each attribute), a name, a type, a long name (for an almost natural language user interface), the unit of measure, a set of legal values that the attribute can have, a definition of the attribute, an indication if the attribute can hold several values at the same time (multivalued) and an indication if the values of the attribute can be supplied by the user (askable). For example, here is an attribute with all of its characteristics.

Attribute number:	1
Name:	interest_rate
Type:	real
Long name:	the current interest rate
Legal values:	0 .. 25
Units:	%
Definition:	The current interest rate is the prime lending rate of interest charged by the bank for a loan
Multivalued:	no
Askable:	yes

The description of an attribute is declared in the KB using five different declaration statements to optimize execution. The syntax of each of these declarations is described below.

(1) The attribute declaration

This declaration defines the attribute number, name, type, long name and unit of measure of the attribute. The types integer, real, and string are predeclared; all others must be declared by the user.

Syntax:

es_attribute(<attribute no>,<name>,<type>,<long name>,<unit>)

Example:

```
es_attribute(1,"interest_rate","integer","current interest
             rate","%")
es_attribute(2,"weather","string","weather conditions","")
```

(2) The legal values declaration

This declaration defines the set of legal values that can be assigned to an attribute. Legal values must be declared for all attributes with the type integer, real or string.

Syntax:

```
legal_values(<attribute no>,<type oflist>([<value>,<value>,...]))
```

where

<type of list> is one of real_list, int_list or str_list.

Example:

```
legal_values(1,real_list([0,25]))
legal_values(4,str_list(["yes","no","maybe"]))
```

Note: When the attribute is of type integer or real, GIEES expects two values that are interpreted as the lower and upper bounds of the interval of legal values.

(3) The attribute definition declaration

This declaration defines the text that will be used to help the user when he is uncertain about the meaning of a question regarding an attribute. It is best to limit each line of text to 50 characters so the screen output will display clearly.

Syntax:

```
attribute_definition(<attribute number>,
                     [<line of text>, <line of text>, ...])
```

Example:

```
attribute definition (1,
   ["the current interest is the prime lending",
    "rate of interest charged by the bank for a loan"])
```

(4) The multivalued declaration

This declaration informs GIEES that this attribute can possess more than one value at the same time.

Syntax:

multivalued([<attribute number>, <attribute number>, ...])

Example:

multivalued([3,4,5])

(5) The askable declaration

This declaration tells GIEES that the value of the associated attribute can be furnished by the user when no rule can be used to infer a value for it.

Syntax:

askable([<attribute number>,<attribute number>, ...])

Example:

askable([2,4,10])

3.4 Rule Description

A rule is described using three types of declarations: rule, condition and conclusion.

(1) The rule declaration

The rule declaration is used to declare a new rule in the KB. It has two parameters to define the rule number and the rule priority. At present, the rule priority is not used by the system, but is included for future uses.

Syntax:

rule(<rule number>,<rule priority>)

Example:

rule (10,3)

(2) The condition declaration

This declaration is used to define the conditions that must be evaluated to verify a rule. A rule can have any number of conditions. A condition is expressed by a statement. A statement is formed by an attribute, a value, and a relation between the attribute and the value. The legal relations for GIEES are

- equal
- not_equal
- less
- less_or_equal
- greater
- greater_or_equal
- in
- not_in
- within_range

Syntax:

```
condition(<rule no>,<attribute no>,<relation>,<value>)
```

where

```
<value> = <type of value>(<value>)
<type of value> = str, int, rel, str_list, int_list,
                  real_list
```

Example:

```
condition(1,3,equal,str("no"))
condition(3,7,in,str_list(["low","medium","high","very_high"]))
```

(3) The conclusion declaration

This declaration informs the IE what fact to infer when the premises of a rule have been verified. This declaration has four parameters: a rule number, an attribute number, a value to be assigned to the attribute, and the conclusion CF for that attribute.

Syntax:

```
conclusion(<rule no>,<attribute no>,<value>,<CF>)
```

where

<value> has the same description as in the condition declaration with the following additions:

```
sum(<object>,<object>,...)  Summation of all objects;
mul(<object>,<object>,...)  Multiplication of all objects;
sub(<object1>,<object2>)    Object2 subtracted from object1;
r_div(<object1>,<object2>)  Object2 divided by object1.
```

and where <object> is either the name of an attribute of type integer, real, or number (constant).

Example:

```
conclusion(2,4,str("snow"),80)
conclusion(2,4,str("rain"),10)
conclusion(2,4,str("sunny"),2)
conclusion(5,4,sum([4,"exterior_temperature"])
conclusion(10,5,mul([1.10,"interest_rate","loan_amount"])
```

3.5 Meta Rules and Facts

This section describes the declaration of base facts, meta rules and other information relevant to the KB.

(1) The user can declare a name for the KB. This name will be used by GIEES for headers, etc. The default name is "No name".

Syntax:

```
kb_name("<string>")
```

Example:

```
kb_name("Mining Expert")
```

(2) In evaluation of a rule, a statement is considered true if the attribute has a CF value of 15 or more. This is the default value which can be changed by the user using the following declaration:

Syntax:

```
truth_minimum(<minimum truth CF>)
```

Example:

```
truth_minimum(20)
```

(3) Often many attributes will have the same set of legal values. Instead of defining a legal_values declaration for each of these attributes, the user can define a type that represents a set of legal values. Consequently an attribute with a user defined type do not have a legal_values declaration.

Syntax:

```
type(<type name>, [<value>,<value>,...])
```

where <value> is a string.

Example:

```
type(boolean,["true","false")
type(scale,["low","medium","high"])
```

(4) The next declaration is used to declare the (default) main goals that GIEES must try to infer. The only parameter is the name of an attribute that must be declared in the attribute declaration section.

Syntax:

```
giees_goal(<attribute name>)
```

Example:

```
giees_goal("crop_condition")
giees_goal("financial_forecast")
```

(5) The fact section is used to declare facts which will always be true in every context. The syntax of the base fact is the same as an infered fact except that the second to last parameter is always constant (-1) indicating a base fact.

Syntax:

```
fact(<attribute no>,<value>,[-1],<CF>)
```

Example:

```
fact(1,"slov_burning",[-1],100)
fact(4,"fast_burning",[-1].90)
fact(8,"inflammable",[-1],80)
```

3.6 Example Of A Knowledge Base

```
/* Declaration Section */
kb_name("Wine Expert")
truth_minimum(20)
type(boolean,["true","false"])
giees_goal("wine")

/* Attribute Section */
es_attribute(1,"wine","string","recommended wine","")
es_attribute(2,"main_course","string","main course","")
es_attribute(3,"sauce","string","type of sauce","")
attribute_definition(2,["The main course is the principal dish"])

legal_values(1,["Entre deux Mers","Beaujolais"])
legal_values(2,["fish","beef"])
legal_values(3,["delicate","spicy"])

askable([2,3])

multivalued([1])

/* Rule Section */
rule(1,0)
rule(2,0)

condition(1,2,"equal","fish")
condition(1,3,"equal","delicate")
condition(2,2,"equal","beef")
condition(2,3,"equal","spicy")
conclusion(1,1,"Entre deux Mers",80)
conclusion(2,1,"Beaujolais",90)
```

4.0 APPLICATIONS

GIEES has been used to develop a prototype knowledge-based system to assist in selection of excavating equipment (Hadjigeorgiou 1988). No other software tools were required. GIEES provided the functions required to create and implement the knowledge bases (i.e. in the form of if-then rules) and the user prompts for information concerning the excavating situation of interest. This application shows that, while GIEES does not offer all of the features of a commercial expert system shell, it can be applied by knowledgeable users to develop knowledge-based or expert systems.

5.0 FUTURE PLANS

Aside from supporting specific expert system applications, the long-term plan is to integrate GIEES with other model-based modules developed in the laboratory to form so-called decision support systems (Nenonen et al. 1988). The plan is to utilize GIEES functions at various points in the decision support system software and to use it to manage the rule sets from one rule-base management interface. An earlier version of GIEES has been implemented using MPROLOG (Logicware 1985). It may be necessary to implement GIEES in yet another dialect of PROLOG to allow implementation of the decision support system on other platforms and to facilitate the linkage of GIEES elements to modules written in C or FORTRAN. Development, refinement and application of AI techniques and languages is expected to feature relatively rapid change that may even lead back to entirely procedural-language implementations to maximize speed (Computer Design 1988). Furthermore, modules based on optimization algorithms either from other sources or developed as required by the laboratory are to be integrated into the decision support systems. The intention is to use GIEES to implement user guidance systems for each of these modules and for the overall decision support system.

6.0 BIBLIOGRAPHY

BORLAND. Turbo PROLOG Reference Manual, Borland Inc.

COMPUTER DESIGN. 1988. AI Techniques Enter the Realm of Conventional Languages. Computer Design, 27(19): 45-49.

HADJIGEORGIOU, J. 1988. A Knowledge-Based System To Aid Selection of Excavating Equipment. Division of Mechanical Engineering, National Research Council Canada, Ottawa, Ont. TR-SYS-018 NRC No. 29529.

HOLSAPPLE, C.W. 1986. Manager's Guide to Expert Systems Using GURU. Dow Jones-Irwin, Homewood, Ill.

JACKSON, P. 1986. Introduction to Expert Systems. Addison-Wesley

LOGICWARE 1985. MPROLOG Documentation. Release 2.1. Logicware Inc., Toronto, Ont.

MARCUS, C. 1986. PROLOG Programming, Addison-Wesley.

M.1 DEMONSTRATION MATERIALS. User's Guide, Teknowledge Inc.

NENONEN, L.K., SCOBLE, M., and HADJIGEORGIOU, J. 1988. Development of a Knowledge-Based Decision Support System for Surface Mining. Computer Applications in the Mineral Industry. Proceedings of the First Canadian Conference on Computers in the Mineral Industry, Quebec, Que. Edited by: K. Fytas, J.L. Collins, and R.J. Singhal. A.A. Balkema, Rotterdam. pp. 587-594.

SCHILDT, H. 1987. Advanced Turbo PROLOG. Osborne McGraw-Hill, Berkeley, CA.

SCHMUCKER, K.J.1984. Fuzzy Sets, Natural Language Computations and Risk Analysis. Computer Science Press, Rockville, MD.

THOMPSON, B.W. 1985. Inside an Expert System. Byte, 10(4),

THOMPSON, B.W. 1986. Finding Rules in Data, Byte, 11(12),

REPORT DOCUMENTATION PAGE/PAGE DE DOCUMENTATION DE RAPPORT

REPORT/RAPPORT TR-SYS-019 1a	REPORT/RAPPORT NRC No. 29977 1b
SECURITY CLASSIFICATION/ CLASSIFICATION DE SÉCURITÉ Unclassified 2	DISTRIBUTION/DIFFUSION ☐ Controlled/Contrôlé ☑ Unlimited/Illimité 3

TITLE/SUBTITLE/TITRE/SOUS-TITRE

General Inference Engine for Expert Systems: User's Guide

4

AUTHOR(S)/AUTEUR(S)

S. Moran

5

SERIES/SÉRIE

6

CORPORATE AUTHOR/PERFORMING AGENCY/AUTEUR D'ENTREPRISE/AGENCE D'EXÉCUTION

7

SPONSORING AGENCY/AGENCE DE SUBVENTION

8

DATE	FILE/DOSSIER	LAB. ORDER COMMANDE DU LAB.	PAGES	DIAGS	REFS
1988/11 9	10	11	20 12a	12b	13 12c

NOTES

13

DESCRIPTORS(KEY WORDS)/MOTS-CLÉS

Inference engine, Expert System shell

14

SUMMARY/SOMMAIRE

A guide for use of a PROLOG-based inference engine and associated rule base manager.

15

ADDRESS/ADRESSE

Dr. Vince Thomson
Head, Systems Laboratory
Division of Mechanical Engineering
National Research Council Canada
Montreal Road, Ottawa, Ontario K1A 0R6
Tel: (613) 993-9461

16